Proc. 6th TAMC, LNCS 5532, pp. 39–48, 2009.

# Fixed-Parameter Algorithms for Graph-Modeled Data Clustering

Jiong Guo

Institut für Informatik, Friedrich-Schiller-Universität Jena
Ernst-Abbe-Platz 2, D-07743 Jena, Germany
jiong.guo@uni-jena.de

**Abstract.** We survey some practical techniques for designing fixed-parameter algorithms for NP-hard graph-modeled data clustering problems. Such clustering problems ask to modify a given graph into a union of dense subgraphs. In particular, we discuss (polynomial-time) kernelizations and depth-bounded search trees and provide concrete applications of these techniques. After that, we shortly review the use of two further algorithmic techniques, iterative compression and average parameterization, applied to graph-modeled data clustering. Finally, we address some challenges for future research.

## 1 Introduction

Clustering a given set of objects according to a given similarity or distance measure requires to partition the input objects into homogeneous and well-separated subsets. This problem finds applications in many areas of computational biology, such as analyzing gene expression [22, 23], proteins [19], gene networks [24], etc. Using graph vertices to represent objects—such as genes or proteins—and adding edges between two vertices iff the interrelation between the two corresponding objects exceeds some threshold value, a clustering with respect to such a graph is a disjoint union of dense subgraphs, also called clusters, such that there are few or no edges between the clusters. A graph is dense if the vertices in it are highly connected. Here, a dense group represents a set of highly similar objects. When formulated as a graph modification problem, one thus asks for a minimum-cardinality set of edge modifications such that in the resulting graph every connected component is a cluster:

> $\Pi$-Cluster Editing:
> **Input**: An undirected graph $G = (V, E)$, a density measure $\Pi$, and an integer $k \geq 0$.
> **Task**: Decide whether there is a set of at most $k$ edge modifications (insertions or deletions) to transform $G$ into a $\Pi$-cluster graph, that is, a graph in which every connected component satisfies $\Pi$.

If the given objects admit a perfect cluster structure, namely, they form well-separated groups and, inside each group, the objects are highly interrelated,

then the graph $G$ should be a disjoint union of dense subgraphs. However, for biological data, the input graph $G$ usually is corrupted and we have to "correct" it under the parsimony criterion to reconstruct the "best" clustering. This is exactly what $\Pi$-Cluster Editing asks for.

One of the most prominent problems in this context is the NP-hard Cluster Editing problem (also known as Correlation Clustering) [3, 23], where the required density measure is $\Pi :=$"being a clique". Cluster Editing has been intensively studied from the viewpoints of polynomial-time approximability as well as parameterized algorithmics [2, 5, 13]. Since the density requirement of "being a clique" has been often criticized for its overly restrictive nature and modeling disadvantages [9, 21], several other natural relaxations of cliques have been considered in the literature, including $s$-defective cliques [25], $s$-plexes [21], $\mu$-cliques [1], etc. Already Cluster Editing being NP-hard [3, 23], we cannot expect algorithms that can solve $\Pi$-Cluster Editing for more general $\Pi$'s efficiently. In practice, heuristic algorithms, approximation algorithms or similar techniques have been employed to cope with the computational intractability of NP-hard problems. However, there are many scenarios where the aforementioned techniques cannot provide satisfactory solutions. For instance, they typically fail when an optimal solution of a computationally hard problem is sought and the solving algorithm should be reasonably efficient. Here, *fixed-parameter algorithms* come into play, where one basically asks for exact algorithms whose running time is exponential only with respect to a certain parameter $k$; in our setting $k$ denotes the number of modification operations. Thus, a $\Pi$-Cluster Editing problem is called *fixed-parameter tractable (FPT)* if it can be solved in $f(k) \cdot |V|^{O(1)}$ time [20].

In this survey, several practically relevant techniques for designing fixed-parameter algorithms for $\Pi$-Cluster Editing are addressed, in particular, *kernelization* (polynomial-time data reduction with provable performance guarantee) and *depth-bounded search trees* that are based on *forbidden subgraph characterizations*. Later, we briefly review two further algorithmic techniques and conclude with some directions for future research.

*Basic graph notations.* We only consider *undirected* graphs $G = (V, E)$, where $n := |V|$ and $m := |E|$. The *(open) neighborhood* $N(v)$ of a vertex $v \in V$ is the set of vertices that are adjacent to $v$ in $G$ and $N^2(v)$ is the set of vertices that have distance exactly two to $v$. The *degree* of a vertex $v$, denoted by $\deg(v)$, is the cardinality of $N(v)$. For a set $U$ of vertices, $N(U) := \bigcup_{v \in U} N(v) \setminus U$. We use $N[v]$ to denote the *closed* neighborhood of $v$, that is, $N[v] := N(v) \cup \{v\}$. For a set of vertices $V' \subseteq V$, the *induced subgraph* $G[V']$ is the graph over the vertex set $V'$ with edge set $\{\{v, w\} \in E \mid v, w \in V'\}$. For $V' \subseteq V$, we use $G - V'$ as an abbreviation for $G[V \setminus V']$ and for a vertex $v \in V$ let $G - v$ denote $G - \{v\}$.

## 2 Kernelizations

To solve NP-hard problems, polynomial-time preprocessing is a natural and promising approach. Preprocessing is based on data reduction techniques that

take a problem's input instance and try to perform a reduction to a smaller, equivalent problem instance, called *reduced* instance. Many practical examples have demonstrated the usefulness of the data reduction techniques [15, 17]. Data reduction and problem kernelization results can explain, and *prove*, why preprocessing works so well in many practical applications. The idea is to prove upper bounds on the size of reduced problem instances, which are then called *problem kernels*. These upper bounds shall be functions solely depending on a parameter that typically is related with the size of a solution set of the problem under study. The formal definition of problem kernels and kernelization reads as follows: Let $(G, k)$ be an instance of a parameterized problem where $k$ denotes the parameter. A *reduction to a problem kernel* (or *kernelization*) means to replace $(G, k)$ by a *reduced* instance $(G', k')$ called *problem kernel* in polynomial time such that (1) $k' \leq k$, (2) $|G'| \leq g(k)$ for some function $g$ only depending on $k$, and (3) $(G, k)$ is a yes-instance if and only if $(G', k')$ is a yes-instance.

In the following, we give two concrete applications of kernelization techniques.

*General Data Reduction Rules and Average-s-Plex Editing* First, we present two general data reduction rules that apply to $\Pi$-Cluster Editing for all considered density measures $\Pi$ but do not necessarily give a problem kernel. Then, we show by the example of Average-$s$-Plex Editing that these rules, however, can lead to fixed-parameter tractability results.

**General rules.** The following rule is clearly true and can be carried out in polynomial time:

**Rule 1** *Remove connected components that satisfy $\Pi$ from $G$.*

The second rule is based on the easy-to-see observation that, if there are more than $k$ edge-disjoint paths between two vertices $u$ and $v$, then $u$ and $v$ must end up in the same $\Pi$-cluster after performing at most $k$ edge modifications to the input graph $G$. Then, we can merge $u$ and $v$ into a "super-vertex". To describe this rule, we introduce weight functions for the vertices and edges. For each super-vertex $\mathcal{X} \in V$, we define a set $V_{\mathcal{X}}$ and two weight functions $\sigma(\mathcal{X})$ and $\delta(\mathcal{X})$: $V_{\mathcal{X}}$ denotes the set of vertices merged into $\mathcal{X}$, $\sigma(\mathcal{X}) := |V_{\mathcal{X}}|$, and $\delta(\mathcal{X})$ is equal to the number of edges between the vertices in $V_{\mathcal{X}}$. Note that, for each of the original vertices $u$ in $V$, we can set $V_u = \{u\}$, $\sigma(u) := 1$ and $\delta(u) := 0$. Moreover, for an edge $e$ between two vertices $u$ and $v$, we define $\omega(e)$ to be the number of the edges between $V_u$ and $V_v$. The next rule reads as follows:

**Rule 2** *If $G$ contains two vertices $u$ and $v$ such that $\omega(\{u, v\}) > k$ or $u$ and $v$ have more than $k$ common neighbors, then remove $u$ from $G$ and set*

- $\sigma(v) := \sigma(u) + \sigma(v)$,
- $\delta(v) := \delta(u) + \delta(v) + \omega(\{u, v\})$, *and*
- $\omega(\{v, w\}) := \omega(\{v, w\}) + \omega(\{u, w\})$ *for each* $w \in V$.

Observe that this rule does not strictly follow the definition of data reduction rules. Its application to an instance of $\Pi$-Cluster Editing results in an instance of some kind of weighted version of $\Pi$-Cluster Editing. However, this

rule may not only be very useful in practice, but also leads to fixed-parameter tractability results, as in the case of Average-$s$-Plex Editing.

**Average-$s$-Plex Editing.** A connected graph $H = (V_H, E_H)$ is an *average-$s$-plex* if the average degree of $H$ is at least $|V_H| - s$. In Average-$s$-Plex Editing, we are given an undirected graph $G = (V, E)$ and an integer $k \geq 0$, and the task is to decide whether it is possible to transform $G$ into a vertex-disjoint union of average-$s$-plexes for a fixed integer $s \geq 1$. It is not hard to see that Rule 2 can be applied to Average-$s$-Plex Editing and results in an instance of a weighted version of this problem. This weighted version is defined as follows.

Extend the function $\sigma(\mathcal{X})$ for super-vertices $\mathcal{X}$ to vertex sets $S$, namely, $\sigma(S) := \sum_{v \in S} \sigma(v)$. The average degree $\bar{d}(G)$ of a connected graph $G = (V, E)$ with vertex weights and edge weights is defined as follows:

$$\bar{d}(G) = \frac{2 \sum_{v \in V} \delta(v) + \sum_{v \in V} \sum_{u \in N(v)} \omega(\{u, v\})}{\sigma(V)}.$$

We say then that a graph is an average-$s$-plex graph if for every connected component $C = (V_C, E_C)$ the average degree $\bar{d}(C)$ is at least $\sigma(V_C) - s$. The weighted version of Average-$s$-Plex Editing, called Weighted Average-$s$-Plex Editing, is defined as follows: Given a graph $G = (V, E)$ with two vertex weight functions $\sigma : V \to [1, n]$ and $\delta : V \to [0, n^2]$, an edge weight function $\omega : E \to [1, n^2]$, and a nonnegative integer $k$, one asks whether there is a set of edge modifications $S$ with $|S| \leq k$ such that applying $S$ to $G$ yields an average-$s$-plex graph.

For such a weighted graph $G = (V, E)$, we have four kinds of edge modifications: we can increase $\omega(\{u, v\})$ by one for an edge $\{u, v\} \in E$, decrease $\omega(\{u, v\})$ by one for an edge $\{u, v\} \in E$, delete $\{u, v\} \in E$ with $\omega(\{u, v\}) = 1$, and add some edge $\{u, v\}$ to $E$ and set $\omega(\{u, v\}) := 1$. Each of these edge operations has cost one, and the overall cost of an edge modification set $S$ is thus exactly $|S|$. We can easily reduce an instance $(G = (V, E), k)$ of Average-$s$-Plex Editing to an instance of the weighted version, by setting $\sigma(v) := 1$ and $\delta(v) := 0$ for each $v \in V$, and $\omega(e) := 1$ for each edge $e \in E$. Note that this reduction is parameter-preserving, that is, $s$ and $k$ are not changed.

Clearly, Rule 2 is a data reduction rule for Weighted Average-$s$-Plex Editing, leading to a graph with a bounded number of vertices.

**Theorem 1 ([16]).** *A yes-instance $(G, k)$ of* Weighted Average-$s$-Plex Editing *that is reduced with respect to Rules 1 and 2 contains at most $4k^2 + 8sk$ vertices.*

We obtain a fixed-parameter algorithm for Weighted Average-$s$-Plex Editing as follows. First, we exhaustively apply the reduction rules, which can clearly be done in polynomial time. If the reduced instance contains more than $4k^2 + 8sk$ vertices, then it is a no-instance. Otherwise, we can solve the problem with running time only depending on $s$ and $k$, for example, by brute-force generation of all possible partitions of the graph. The fixed-parameter tractability of Average-$s$-Plex Editing then directly follows from the described reduction to Weighted Average-$s$-Plex Editing.

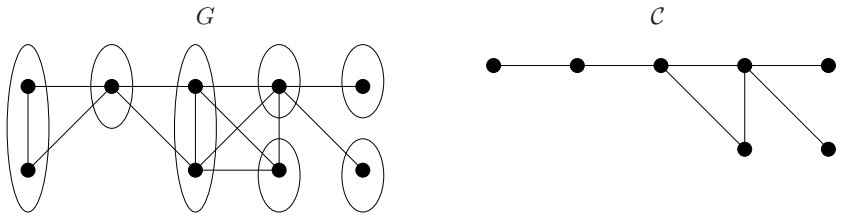


**Fig. 1.** A graph $G$ and its critical clique graph $\mathcal{C}$. *Ovals* denote the critical cliques of $G$.

*Cluster Editing.* As mentioned in Section 1, Cluster Editing is the special case of $\Pi$-Cluster Editing with $\Pi :=$"being a clique". This means that, given a graph $G = (V, E)$ and $k \geq 0$, Cluster Editing asks to decide whether $G$ can be transformed by at most $k$ edge deletions and insertions into a cluster graph, that is, a vertex-disjoint union of cliques. We describe a kernelization for this problem which results in a graph with $O(k)$ vertices [14]. First, we introduce the concepts of *critical clique* and *critical clique graph* .

**Definition 1.** *A* critical clique *of a graph $G$ is a clique $K$ where all vertices of $K$ have the same sets of neighbors in $V \setminus K$, and $K$ is maximal under this property. Given a graph $G = (V, E)$, let $\mathcal{K}$ be the collection of its critical cliques. Then the* critical clique graph $\mathcal{C}$ *is the graph $(\mathcal{K}, E_{\mathcal{C}})$ with $\{K_i, K_j\} \in E_{\mathcal{C}} \iff \forall u \in K_i, v \in K_j : \{u, v\} \in E$. That is, the critical clique graph has the critical cliques as nodes, and two nodes are connected iff the corresponding critical cliques together form a larger clique.*

See Figure 1 for an example of a graph and its critical clique graph. The critical clique graph can be constructed in $O(m+n)$ time [14]. Note that we use the term *nodes* for the vertices in the critical clique graph. Moreover, we use $K(v)$ to denote the critical clique containing vertex $v$ and use $V(K)$ to denote the set of vertices contained in the critical clique $K \in \mathcal{K}$.

The basic idea behind introducing critical cliques is the following: suppose that the input graph $G = (V, E)$ has a solution with at most $k$ edge modifications. Then, at most $2k$ vertices are "affected" by these edge modifications, that is, they are endpoints of edges added or deleted. Thus, in order to give a size bound on $V$ depending only on $k$, it remains to upper-bound the size of the "unaffected" vertices. The central observation is that, in the cluster graph obtained after making the at most $k$ edge modifications, the unaffected vertices contained in one clique must form a critical clique in the original graph $G$. By this observation, it seems easier to derive data reduction rules working for the critical cliques and the critical clique graph than to derive rules directly working on the input graph. Concerning critical cliques, one can show that there is always a solution that does not divide critical cliques. Moreover, large critical cliques, together with their neighbors, must form clusters in the final cluster graph.

**Rule 3** *If a critical clique $K$ contains more vertices than all the critical cliques in $N_{\mathcal{C}}(K) \cup N^2_{\mathcal{C}}(K)$ together, then construct a clique $C$ consisting of $V(K)$ and $V(K')$ for all $K' \in N_{\mathcal{C}}(K)$, remove all vertices in $C$ from $G$, and decrease the parameter $k$ by the number of edge insertions and deletions needed to construct $C$ and to separate $C$ from the rest of $G$.*

Combining Rules 1 and 3 gives the following problem kernel.

**Theorem 2 ([14]).** *If a reduced graph for* Cluster Editing *has more than $6k$ vertices, then it has no solution with at most $k$ edge modifications.*

Independently, Fellows et al. [11] achieved also the $6k$-vertex kernel by using a different approach. By adding a more intricate data reduction rule, the problem kernel size can be improved to at most $4k$ vertices [14].

## 3 Forbidden Subgraph Characterization and Search Tree

The basic idea behind the depth-bounded search tree technique is to organize the systematic and exhaustive exploration of the search space in a tree-like manner. More precisely, given an instance $(G, k)$ of $\Pi$-Cluster Editing, search tree algorithms replace $(G, k)$ by a finite set $\mathcal{C}$ of instances $(G_i, k_i)$ with $1 \leq i \leq |\mathcal{C}|$ and $k_i < k$ specified by some *branching rules*. If $G_i$ for an $i$ is not a $\Pi$-cluster graph, then the algorithm recursively applies this replacing procedure to $(G_i, k_i)$. The algorithm terminates when the replacing procedure is no longer applicable, that is, there is a $G_i$ being a $\Pi$-cluster graph or $k_i < 0$. The recursive application of the replacing procedure can be illustrated in a tree structure, yielding a *search tree*. The depth of the search tree is bounded from above by a function of the parameter and its size is clearly $O(|\mathcal{C}|^k)$.

Many graph properties allow for characterizations in terms of forbidden subgraphs.

**Definition 2.** *Let $\mathcal{F}$ be a collection of graphs. A graph property $\Pi$ can be characterized by the* forbidden induced subgraphs *in $\mathcal{F}$ iff each graph having $\Pi$ contains no graph in $\mathcal{F}$ as an induced subgraph. A graph having $\Pi$ is called* $\mathcal{F}$-free.

It is not hard to observe that *finite* forbidden subgraph characterizations lead to the fixed-parameter tractability of the corresponding graph modification problems: Given a forbidden subgraph $G'$, a branching rule considers all possible ways to destroy $G'$; in each case, an edge is deleted from or added to $G'$. Therefore, the search tree size depends directly on the size of $G'$.

For many density measures $\Pi$, $\Pi$-cluster graphs admit such finite forbidden subgraph characterizations. In the following, we give two applications of the search tree technique that is based on forbidden subgraph characterizations.

*Cluster Editing.* For $\Pi :=$"being a clique", it is easy to observe that the only forbidden subgraph is an induced path of three vertices. An $O(3^k)$-size search tree follows immediately from this characterization: If the graph $G$ is already a union of disjoint cliques, then we are done: Report the solution and return; otherwise, if $k \leq 0$, then we cannot find a solution in this branch of the search tree. Otherwise, identify $u, v, w \in V$ with $\{u, v\} \in E$, $\{u, w\} \in E$, but $\{v, w\} \notin E$. Recursively call the branching procedure on the following three instances consisting of graphs $G' = (V, E')$ with nonnegative integer $k'$: (B1) $E' := E - \{u, v\}$ and $k' := k - 1$, (B2) $E' := E - \{u, w\}$ and $k' := k - 1$, and (B3) $E' := E + \{v, w\}$ and $k' := k - 1$. The search tree size can be significantly reduced. More specifically, a more sophisticated branching strategy gives a search tree size of $O(2.27^k)$ [13]. To this end, we distinguish two cases concerning three vertices $u, v, w \in V$ with $\{u, v\} \in E$, $\{u, w\} \in E$, but $\{v, w\} \notin E$: (C1) Vertices $v$ and $w$ do not share a common neighbor, that is, $\nexists x \in V, x \neq u : \{v, x\} \in E$ and $\{w, x\} \in E$; (C2) Vertices $v$ and $w$ have a common neighbor $x \neq u$.

The key observation for the improvement is the following observation, which implies, regarding case (C1), a branching into two cases (B1) and (B2) suffices.

**Lemma 1 ([13]).** *Given a graph $G = (V, E)$, a nonnegative integer $k$ and $u, v, w \in V$ with $\{u, v\} \in E$, $\{u, w\} \in E$, but $\{v, w\} \notin E$. If $v$ and $w$ do not share a common neighbor besides $u$, then branching case (B3) cannot yield a better solution than both cases (B1) and (B2), and can therefore be omitted.*

For case (C2), the standard branching into three subcases can be avoided by taking a common neighbor of $v$ and $w$ into account [13]. Combining the analysis for cases (C1) and (C2), one can show the following.

**Theorem 3 ([13]).** Cluster Editing *can be solved by a search tree algorithm with search tree size $O(2.27^k)$.*

Note that, by considering more complicated case distinction, Böcker et al. [5] achieved a search tree of size $O(1.82^k)$.

*$s$-Defective Clique Editing.* A connected graph $G = (V, E)$ is an $s$-defective clique if $|E| \geq |V| \cdot (|V| - 1)/2 - s$ for an integer $s \geq 0$. An $s$-defective clique graphs consists of connected components that are $s$-defective cliques. In the following, we present a forbidden subgraph characterization of $s$-defective clique graphs for any $s \geq 0$ as well as a search tree algorithm that makes use of this characterization. First, we show that $s$-defective clique graphs are characterized by forbidden subgraphs with $O(s)$ vertices. Note that, if $s = 0$, then $s$-defective clique graphs are equal to cluster graphs and the only forbidden subgraph is a path induced by three vertices, as shown above. Here, a graph is called a *minimal* forbidden subgraph if all its subgraphs are $s$-defective cliques.

**Theorem 4 ([16]).** *For $s \geq 1$, any minimal forbidden induced subgraph of $s$-defective clique graphs contains at most $2(s + 1)$ vertices.*

The forbidden subgraph characterization given in Theorem 4 directly leads to a search tree algorithm for $s$-Defective Clique Editing: Given a graph $G =$

$(V, E)$ that is not an $s$-defective clique graph, one can find in $O(nm)$ time a minimal forbidden subgraph [16]. Then, branch into all possibilities of adding or deleting an edge between two vertices contained in the forbidden induced subgraph. Since the number of vertices of a minimal forbidden subgraph is bounded by $2s + 2$, there are at most $\binom{2s+2}{2}$ cases and in each case the parameter $k$ is decreased by one. Hence, the size of the search tree is bounded by $O\left(\binom{2s+2}{2}^k\right)$. Putting all together leads to the following.

**Theorem 5 ([16]).** *$s$-Defective Clique Editing is fixed-parameter tractable with respect to the combined parameter $(s, k)$.*

## 4 Further Techniques

*Iterative Compression and Cluster Vertex Deletion.* Cluster Vertex Deletion is the vertex-deletion version of Cluster Editing. The only difference is the modification allowed. In Cluster Vertex Deletion, the only allowed operation is the deletion of vertices. A solution for this problem is called a "cluster vertex deletion set" (CVD, for short). Hüffner et al. [18] showed that the fairly recently introduced *iterative compression* technique can be applied to this problem, resulting in an algorithm running in $O(2^k k^6 \log k + n^3)$ time. The general idea behind their iterative compression is as follows: Given a graph $G = (V, E)$ and $k \geq 0$, start with $V' = \emptyset$ and $X' = \emptyset$; clearly, $X'$ is a CVD for $G[V']$. Iterating over all graph vertices, step by step add one vertex $v \notin V'$ from $V$ to both $V'$ and $X$. Then $X$ is still a CVD set for $G[V']$, although possibly not a minimum one. One can, however, obtain a minimum one by applying the *compression routine*. It takes a graph $G$ and a CVD $X$ for $G$, and returns a minimum CVD for $G$. Therefore, it is a loop invariant that $X$ is a minimum-size CVD for $G[V']$. Since eventually $V' = V$, one obtains an optimal solution for $G$ once the algorithm returns $X$. Together with the following lemma, the correctness and overall running time of the iterative compression algorithm follow.

**Theorem 6 ([18]).** *The compression routine for Cluster Vertex Deletion runs in $O(2^k \cdot m\sqrt{n} \log n)$ time.*

*Average Parameterization and Consensus Clustering.* At the first sight, Consensus Clustering is not a graph-modeled clustering problem. However, one can reformulate it as a weighted version of Cluster Editing as stated in [8]. In Consensus Clustering, one is given a base set $S$ and a set $\mathcal{C}$ of partitions over $S$, and asks for a partition $C$ of $S$ minimizing $\sum_{C_i \in \mathcal{C}} d(C, C_i)$. The distance $d(C, C_i)$ between two partitions is defined as follows: we call two elements $a, b \in S$ *co-clustered* with respect to a partition $C$ if $a$ and $b$ occur together in a subset of $C$ and *anti-clustered* if $a$ and $b$ occur in different subsets of $C$. Define the *distance* $d(C_i, C_j)$ between two partitions $C_i$ and $C_j$ as the number of unordered pairs $\{a, b\}$ of elements from the base set $S$ such that $a$ and $b$ are co-clustered in one of $C_i$ and $C_j$ and anti-clustered in the other. Recently, Consensus Clustering has been shown to be fixed-parameter tractable with respect to the

*average distance between the input partitions* [4] by deriving two data reduction rules and showing the existence of a "pseudo-kernel". By defining the average distance between the input partitions as $d := \big(\sum_{C_i,C_j\in\mathcal{C}} d(C_i,C_j)\big)/\big(n\cdot(n-1)\big)$, one can show the following:

**Theorem 7 ([4]).** *Each* Consensus Clustering *instance can be reduced in polynomial time to an equivalent instance with at most* $9d+4$ *elements.*

## 5 Conclusion

As demonstrated by several experimental studies [6, 10], data reduction proves extremely useful for practically solving hard problems. Even complex and large graphs such as biological instances allow for exact solutions. Therefore, encountering an NP-hard graph clustering problem, one should always start with designing data reduction rules. Even a data reduction that has no provable performance guarantee may turn out to be very effective in practice. Concerning the search tree technique, notice that the running times given here and in the literature are based on worst-case analysis and, thus, often too pessimistic.

There still remain a lot of challenges concerning the design of fixed-parameter algorithms for graph clustering problems: Until now, experimental studies of fixed-parameter algorithms in the field of graph-modeled clustering concentrated on Cluster Editing. There lack efficient implementations for other variants of $\Pi$-Cluster Editing. For example, for $s$-Defective Clique Editing, no polynomial-size kernel is known. A more general version of Cluster Editing considers graphs with *don't care*-edges, that is, edges with zero cost [3]. It is open whether this problem is fixed-parameter tractable [7]. In some practical applications, the dense clusters do not need to be disjoint. the graphs resulting by the edge modifications are not necessarily vertex-disjoint dense clusters; some vertices may be contained in more than one dense clusters. A very first step in this direction has been undertaken recently [12], but this field is widely open.

## References

[1] J. Abello, M. G. C. Resende, and S. Sudarsky. Massive quasi-clique detection. In *Proc. 5th LATIN*, volume 2286 of *LNCS*, pages 598–612. Springer, 2002. 2

[2] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5), 2008. Article No. 23. 2

[3] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1–3):89–113, 2004. 2, 9

[4] N. Betzler, J. Guo, C. Komusiewicz, and R. Niedermeier. Average parameterization for computing medians, 2009. Submitted. 9

[5] S. Böcker, S. Briesemeister, Q. B. A. Bui, and A. Truß. Going weighted: Parameterized algorithms for cluster editing. In *Proc. 2nd COCOA*, volume 5165 of *LNCS*, pages 1–12. Springer, 2008. 2, 7

[6] S. Böcker, S. Briesemeister, and G. W. Klau. Exact algorithms for cluster editing: Evaluation and experiments. In *Proc. 7th WEA*, volume 5038 of *LNCS*, pages 289–302. Springer, 2008. 9

[7] H. L. Bodlaender, M. R. Fellows, P. Heggernes, F. Mancini, C. Papadopoulos, and F. A. Rosamond. Clustering with partial information. In *Proc. 33rd MFCS*, volume 5162 of *LNCS*, pages 144–155. Springer, 2008. 9

[8] P. Bonizzoni, G. D. Vedova, R. Dondi, and T. Jiang. On the approximation of correlation clustering and consensus clustering. *J. Comput. Syst. Sci.*, 74(5):671–696, 2008. 8

[9] E. J. Chesler, L. Lu, S. Shou, Y. Qu, J. Gu, J. Wang, H. C. Hsu, J. D. Mountz, N. E. Baldwin, M. A. Langston, D. W. Threadgill, K. F. Manly, and R. W. Williams. Complex trait analysis of gene expression uncovers polygenic and pleiotropic networks that modulate nervous system function. *Nature Genetics*, 37(3):233–242, 2005. 2

[10] F. K. H. A. Dehne, M. A. Langston, X. Luo, S. Pitre, P. Shaw, and Y. Zhang. The cluster editing problem: Implementations and experiments. In *Proc. 2nd IWPEC*, volume 4169 of *LNCS*, pages 13–24. Springer, 2006. 9

[11] M. R. Fellows, M. A. Langston, F. A. Rosamond, and P. Shaw. Efficient parameterized preprocessing for Cluster Editing. In *Proc. 16th FCT*, volume 4639 of *LNCS*, pages 312–321. Springer, 2007. 6

[12] M. R. Fellows, J. Guo, C. Komusiewicz, R. Niedermeier, and J. Uhlmann. Graph-based data clustering with overlaps, 2009. Submitted. 9

[13] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-modeled data clustering: Exact algorithms for clique generation. *Theory Comput. Syst.*, 38(4):373–392, 2005. 2, 7

[14] J. Guo. A more effective linear kernelization for Cluster Editing. *Theor. Comput. Sci.*, 410(8-10):718–726, 2009. 5, 6

[15] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007. 3

[16] J. Guo, I. A. Kanj, C. Komusiewicz, and J. Uhlmann. Editing graphs into dense clusters, 2009. Submitted. 4, 7, 8

[17] F. Hüffner, R. Niedermeier, and S. Wernicke. Techniques for practical fixed-parameter algorithms. *The Computer Journal*, 51(1):7–25, 2008. 3

[18] F. Hüffner, C. Komusiewicz, H. Moser, and R. Niedermeier. Fixed-parameter algorithms for cluster vertex deletion. *Theory Comput. Syst.*, 2009. To appear. 8

[19] H. Kawaji, Y. Takenaka, and H. Matsuda. Graph-based clustering for finding distant relationships in a large set of protein sequences. *Bioinformatics*, 20(2): 243–252, 2004. 1

[20] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. 2

[21] S. B. Seidman and B. L. Foster. A graph-theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, 6:139–154, 1978. 2

[22] S. Seno, R. Teramoto, Y. Takenaka, and H. Matsuda. A method for clustering expression data based on graph structure. *Genome Informatics*, 15(2):151–160, 2004. 1

[23] R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Appl. Math.*, 144(1–2):173–182, 2004. 1, 2

[24] B. H. Voy, J. A. Scharff, A. D. Perkins, A. M. Saxton, B. Borate, E. J. Chesler, L. K. Branstetter, and M. A. Langston. Extracting gene networks for low-dose radiation using grpah theoretical algorithms. *PLoS Computational Biology*, 2(7): e89, 2006. 1

[25] H. Yu, A. Paccanaro, V. Trifonov, and M. Gerstein. Predicting interactions in protein networks by completing defective cliques. *Bioinformatics*, 22(7):823–829, 2006. 2